

# Jess To JADE Tool Kit

A Rule-Based Solution Supporting Intelligent and Adaptive Agents

Joël Vogt  
13.8.2008

University of Fribourg, Switzerland  
Department of Informatics



# Agenda

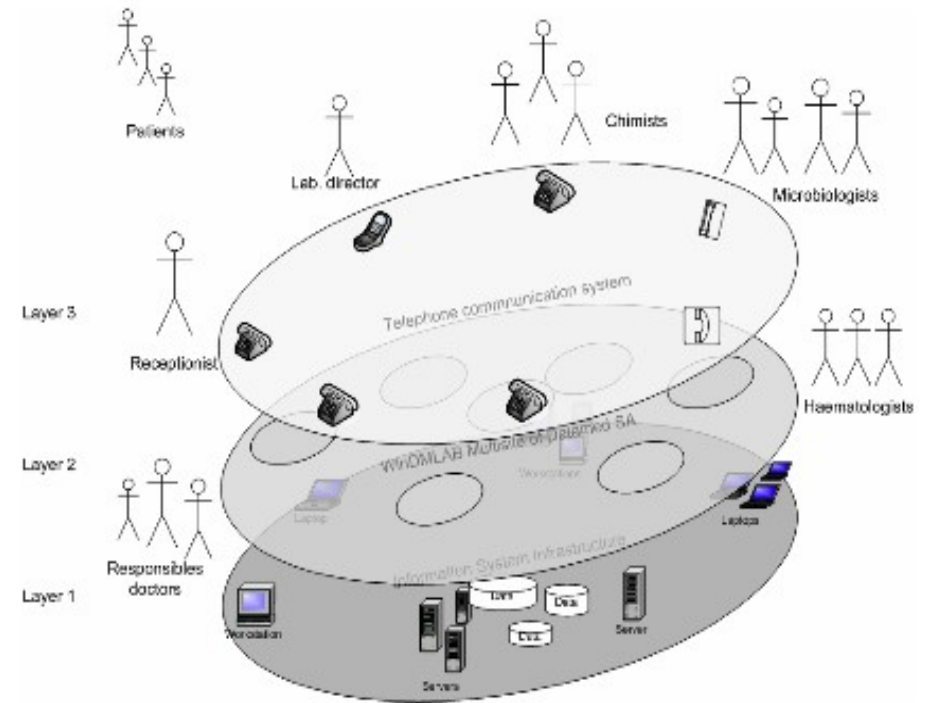
- Goal of this thesis
- Case Study: Cantonal Hospital of Fribourg
- Theoretical Background:
  - Software Agents
  - Rule Engines
- Jess and JADE Integration
- MediMAS Goes Jess
- Conclusion

# Goal of this Thesis

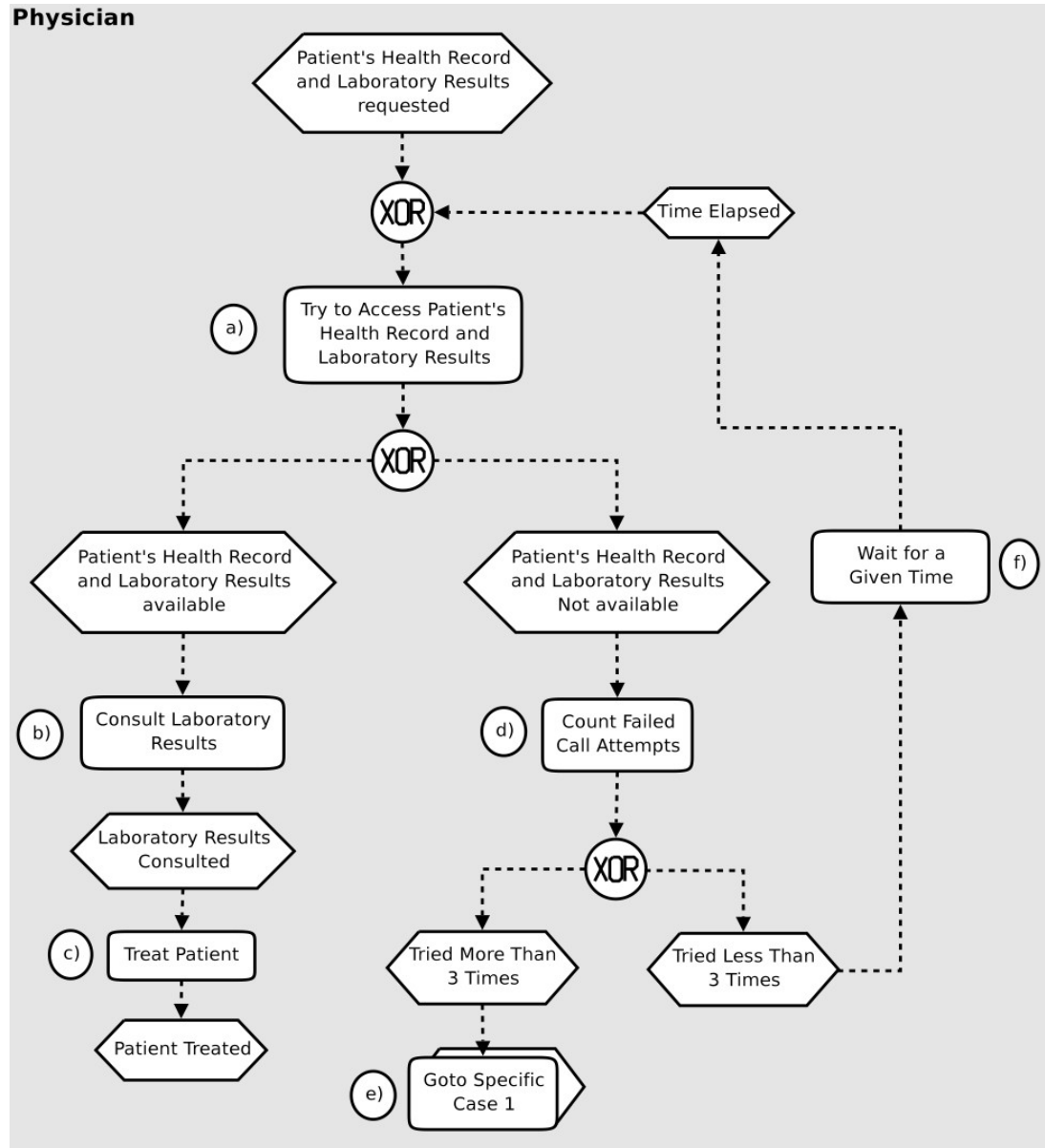
- Integration of Jess and JADE to manage business processes.
- Why?
  - JADE: framework for distributed, scalable, extensible agent architecture
  - Jess permits to: add rules at run time, decides on the execution of the rules, provides interface to access the knowledge base, has event handler interface, etc.

# Case Study: Cantonal Hospital of Fribourg

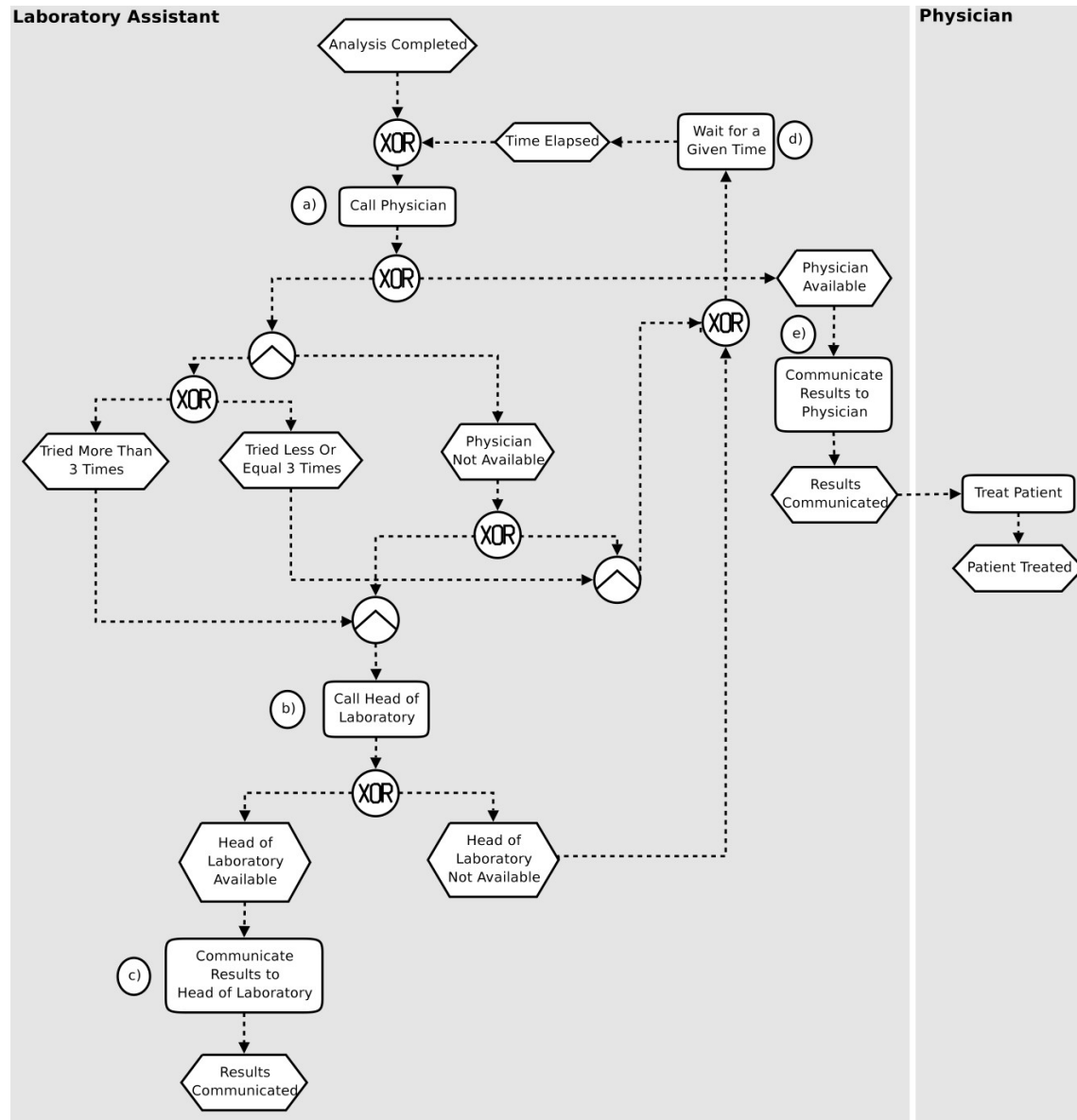
- Current Situation
  - Several actors from different departments
    - WinDmlab
    - Phone
  - Communication with
    - Phone is not logged.
    - What happens when information is critical?



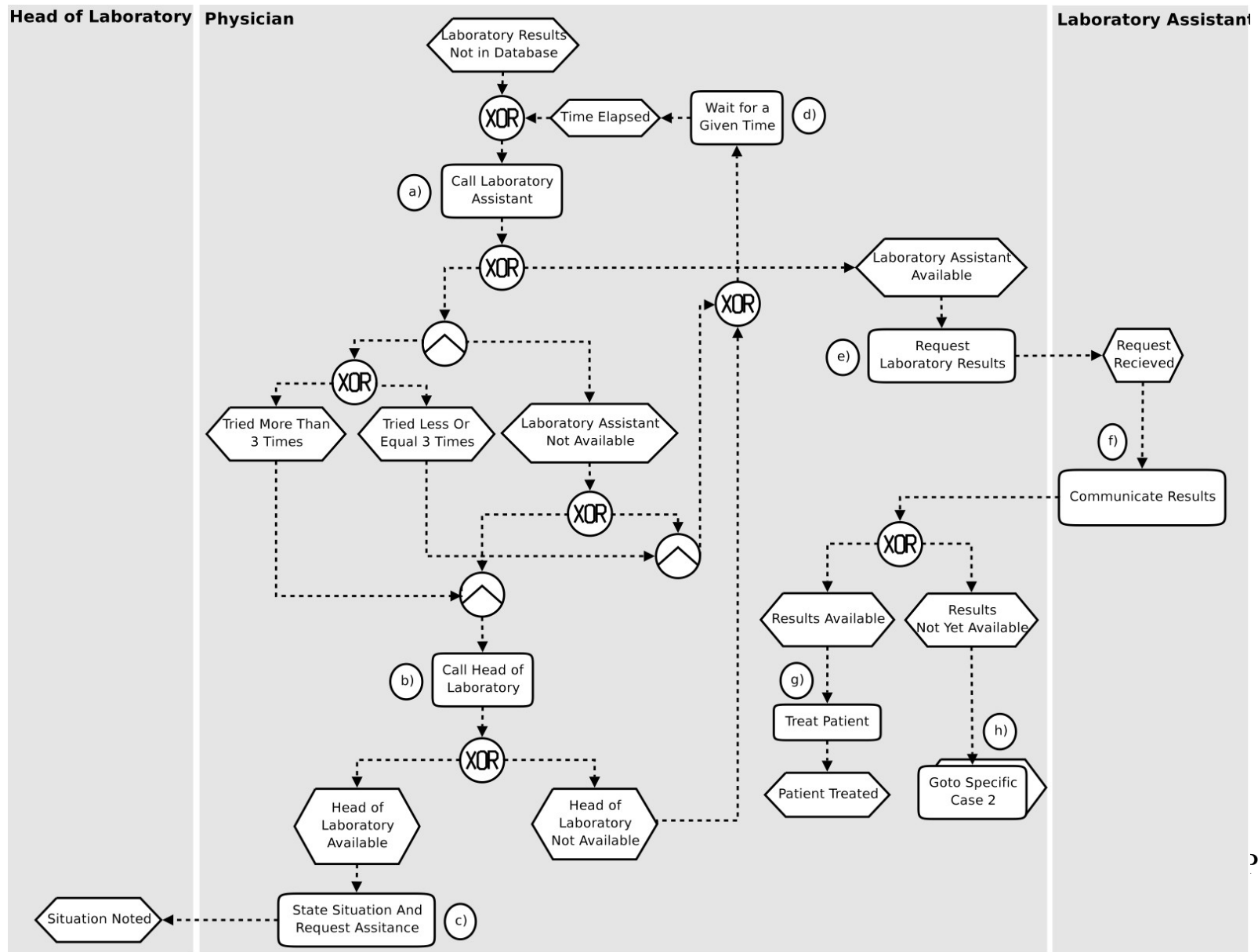
# Case Study: Cantonal Hospital of Fribourg



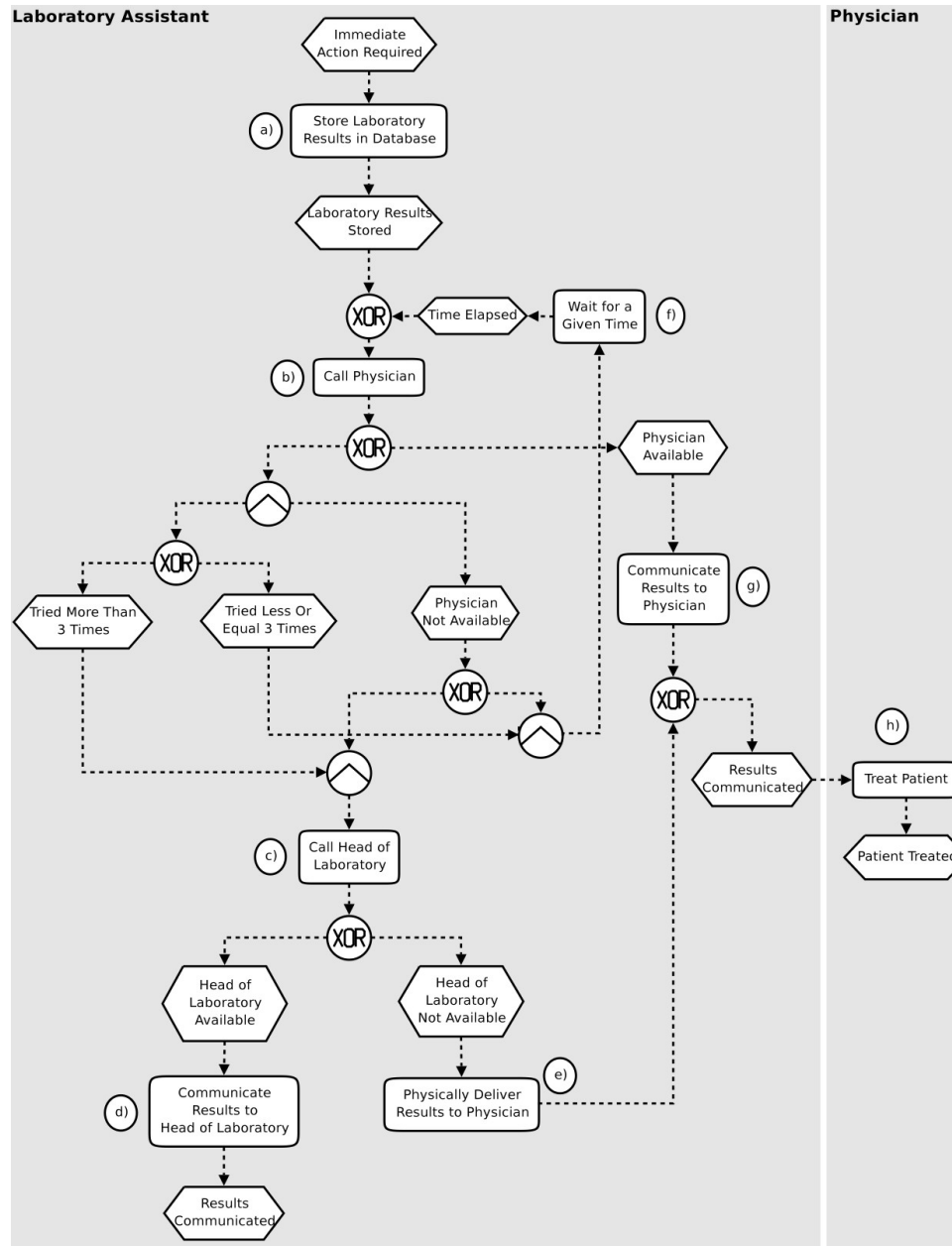
# Case Study: Cantonal Hospital of Fribourg



# Case Study: Cantonal Hospital of Fribourg



# Case Study: Cantonal Hospital of Fribourg



# Case Study: Cantonal Hospital of Fribourg

## •MediMAS

- adds an agent layer on top of the existing infrastructure.
- Every actor is represented by an software agent.
- If human is not available, agents will handle it automatically.

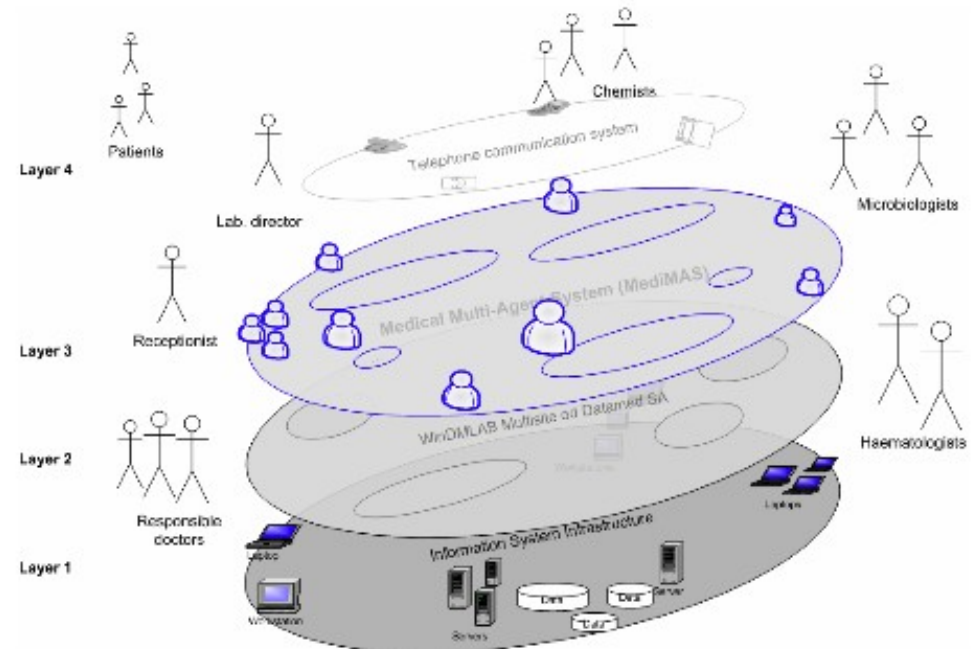
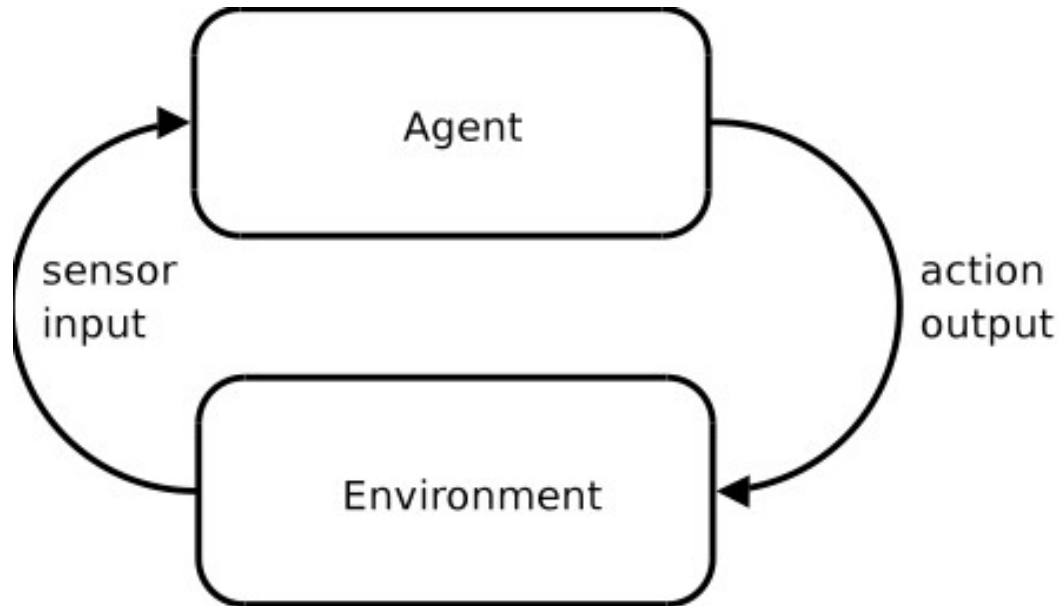


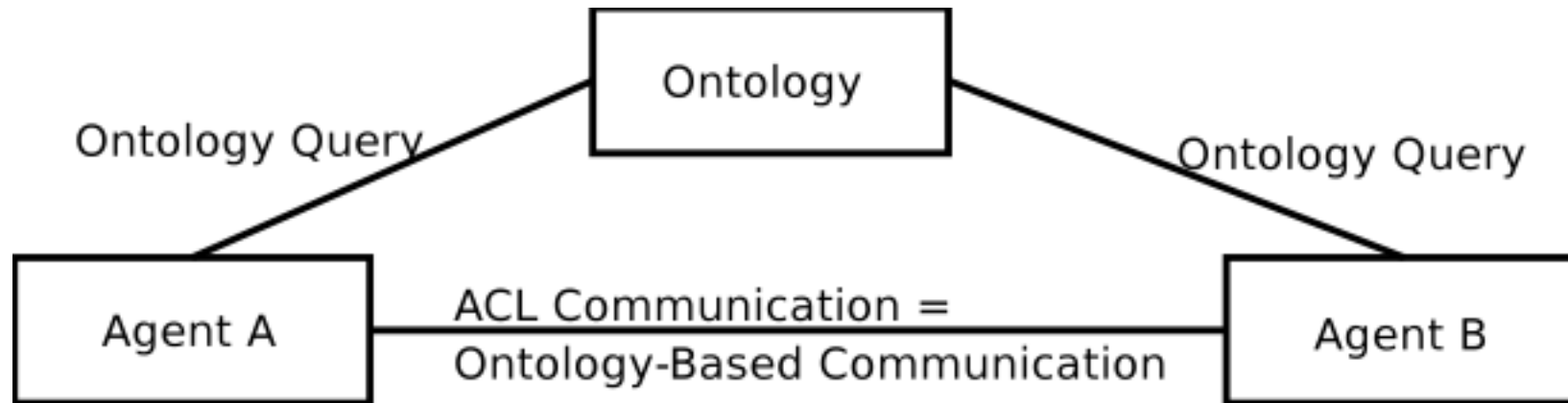
Figure 3. A software agent-based solution

# Software Agents

- Properties:
  - Autonomy
  - Social ability
  - Reactivity
  - Pro-activeness
- Architectures:
  - Reactive
  - Intelligent



# Agent communication



- Definition: "An ontology is an explicit specification of a conceptualization" [Gru93, p. 1].
- Management:
  - Central ontology server.
  - Agents possess the ontology.

# Multi-Agent Framework JADE

- Opensource agent development framework written in Java.
- JADE agents are:
  - autonomous and proactive.
  - loosely coupled.
- JADE agents communicate P2P.
- JADE is fully FIPA compliant.

# Agents Behaviors

- Behaviors contain the actions JADE agents execute.
- An agent may have several behaviors
- Behaviors are like threads.
- Each Behavior is executed one after another.
- Behaviors that react to messages can be provided with filters.

# Containers

- Hosts a set of agents.
- Provides basis for agent communication.
- Has its own Java process.
- Agents are Java threads.

# JADE Ontologies

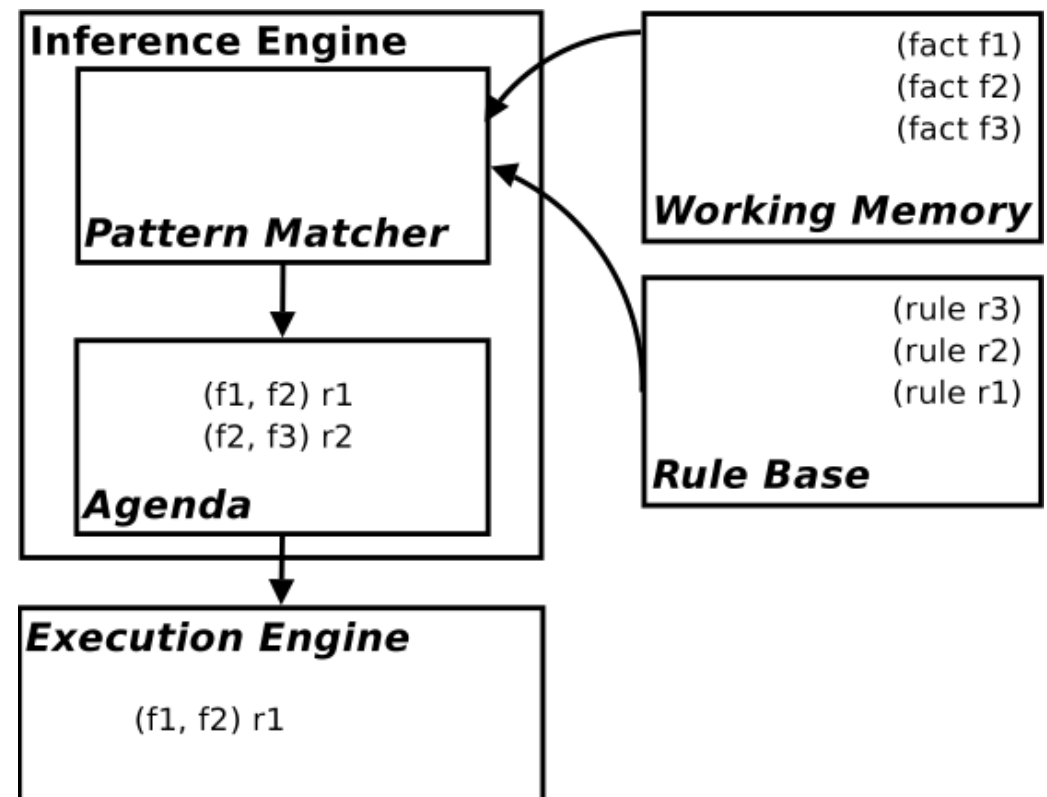
- Represented as a Java object.
- Content manager transforms POJOs into string representation and back.

# Rule-Based Systems

- Knowledge is represented as heuristics, rules apply when a given situation occurs.
- Programmer specifies what not when.
- Comparable to If-Then statements

# Architecture of a Rule-Based System

- Pattern matcher: applies rules to facts
- Agenda: Conflict resolution: decision on which rule is fired.

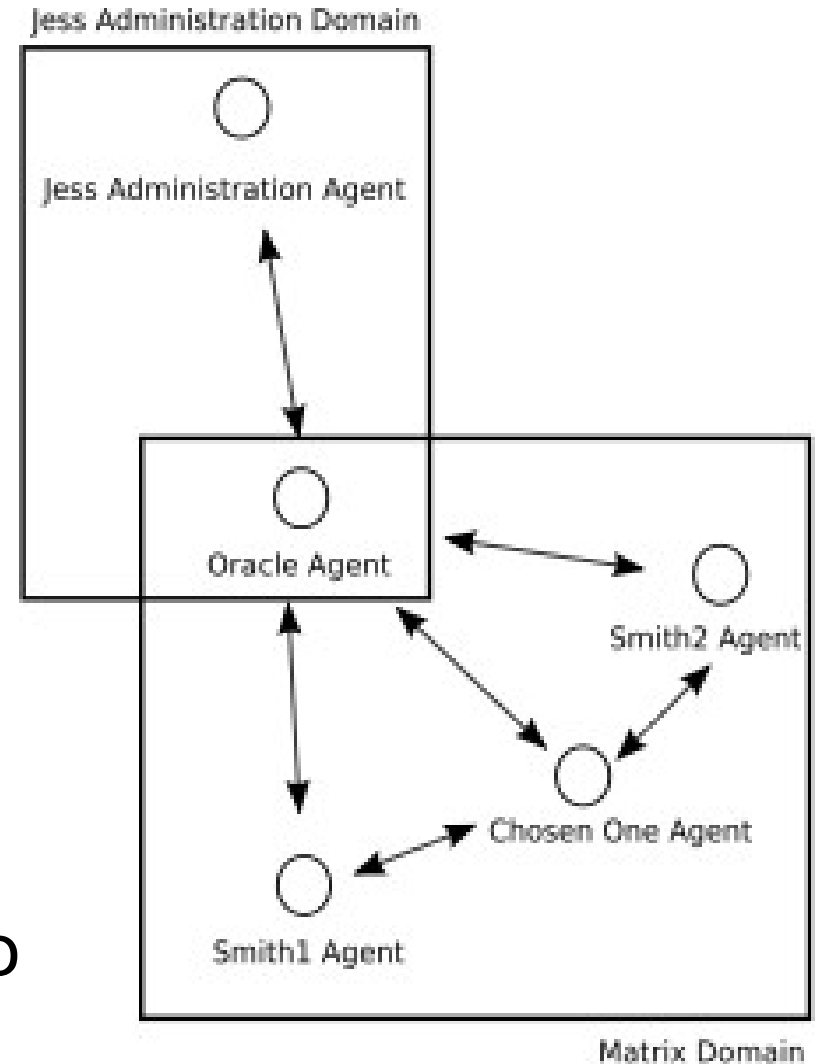


# Jess Rule-Based System

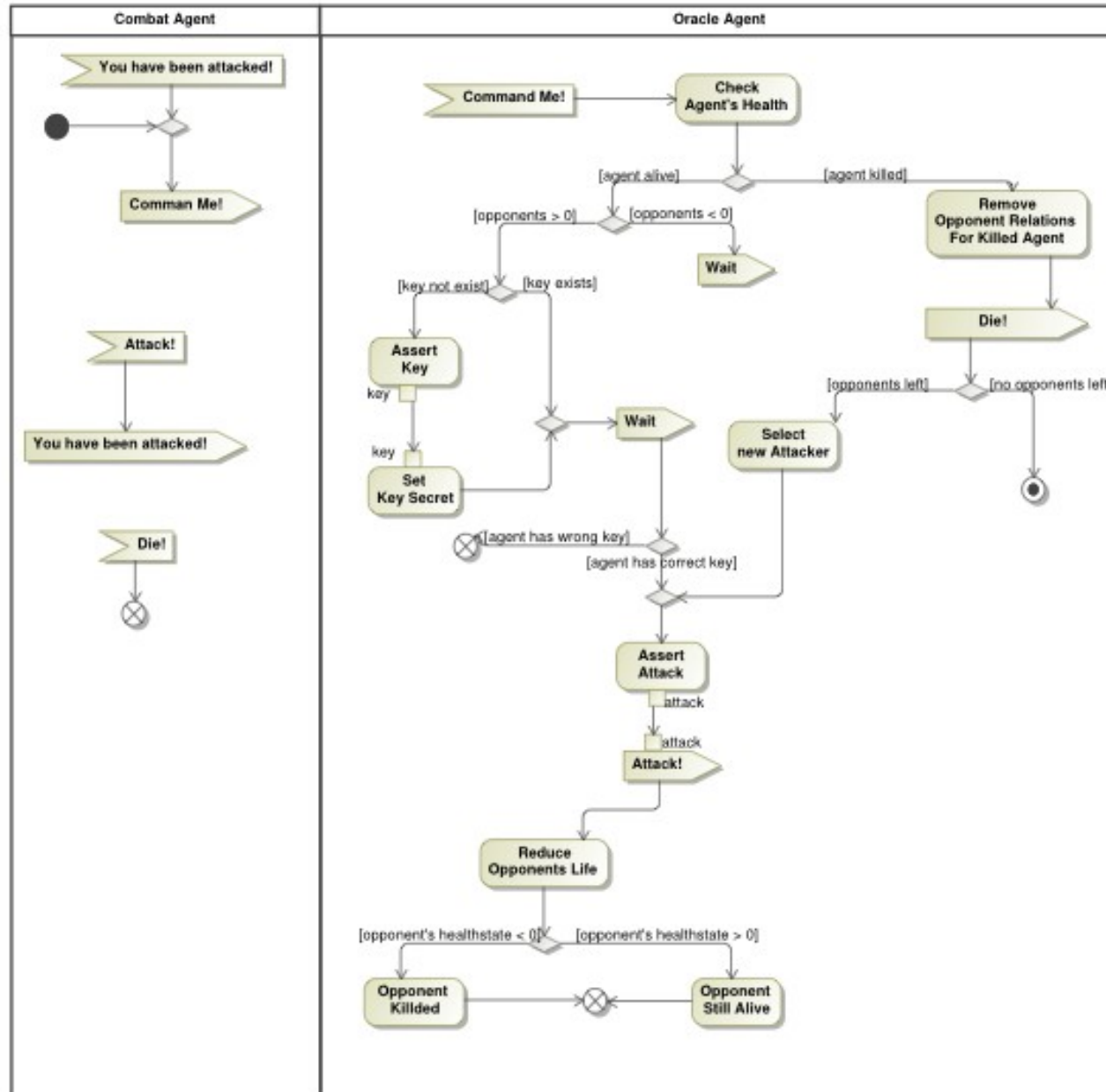
- Rule-engine written entirely in Java, thus fully integrated with Java.
- Jess is an interpreter: rules can be added and removed during runtime.
- Jess is free for academics.

# Jess and JADE Integration: The Matrix Example

- What is the Matrix?
  - An agent environment developed to explore the possibilities with Jess and JADE.
- Organizational structuring:
  - Master agent “Oracle”
  - Slave agents “Smith” and “ChosenOne”
  - Jess Administration agent to administer Jess remotely



# Main Workflow



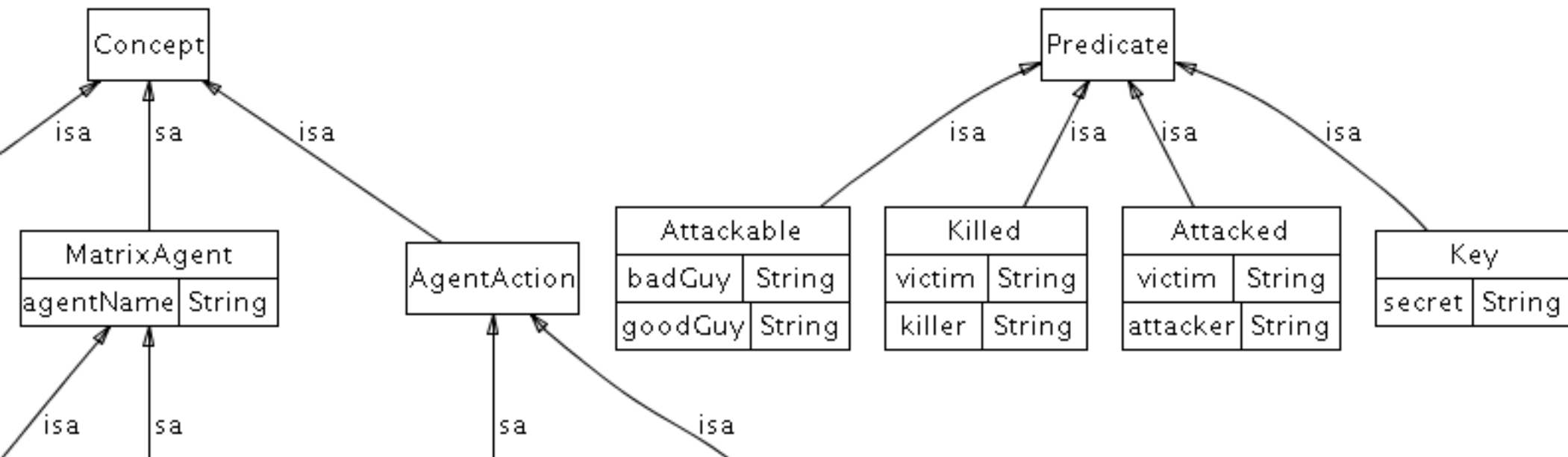
# Rules for the Workflow

```
(defrule AGENT-FATE::agent-alive ;entry point if agent is alive
?cm <- (CommandMe (slave ?s))
?agent <- (CombatAgent (agentName ?s) {healthstate > 0})
=>
(printout t "agent " ?s " is getting a wait" crlf)
(assert (Wait (slave ?s) (secret "")))
(return))
```

```
(defrule AGENT-FATE::agent-dead ;entry point if agent is dead
?cm <- (CommandMe (slave ?s) (secret ?sc))
?agent <- (CombatAgent (agentName ?s) )
(exists (Attackable {badGuy == ?s || goodGuy ==?s}))
(Killed (victim ?s))
=>
(assert (Die (slave ?s) (secret ?sc)))
(focus CLEANUP))
```

```
(defrule AGENT-FATE::assert-key-fact ;only create a key if a fight is possible
(CommandMe (slave ?s))
  ?agent <- (CombatAgent (agentName ?s) (healthstate ?hsc&: (> ?hsc 0)))
  ;(Wait (slave ?s))
(exists (Attackable {badGuy == ?s || goodGuy ==?s}))
(not (Key))
=>
(printout t " asserting key" crlf)
(assert (Key (secret "illuminati"))))
(focus FIRST-KEY-ASSIGNMENT))
```

# The Matrix Ontology



- Specifies sets used by JADE and Jess.
- Both use the same ontology thus they can interoperate seamlessly.

# Agent Conversations

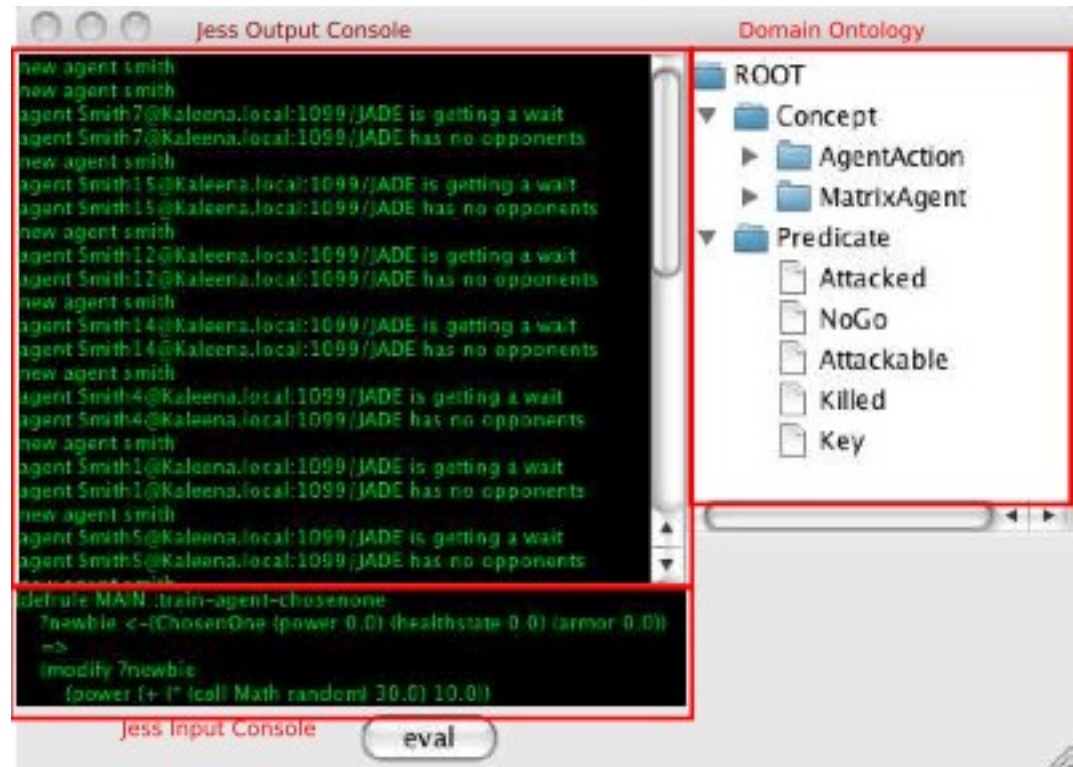
- Agent conversations are fully FIPA compliant
- For each agent action, there is one initiator and one responder behavior.
- E.g. “Attack” agent action has “AttackInitiator” behavior and “AttackResponder” behavior.

# Agent Behaviors

- Two Types:
  - Internal Behaviors: Do not offer an interface to other agents.
  - Communicational Behaviors: Should adhere to standards.
    - All external behaviors extend JADE FIPA-Protocol behavior classes.
  - Concrete behaviors only implement the “business logic”.
  - Dull work is handled by super classes.

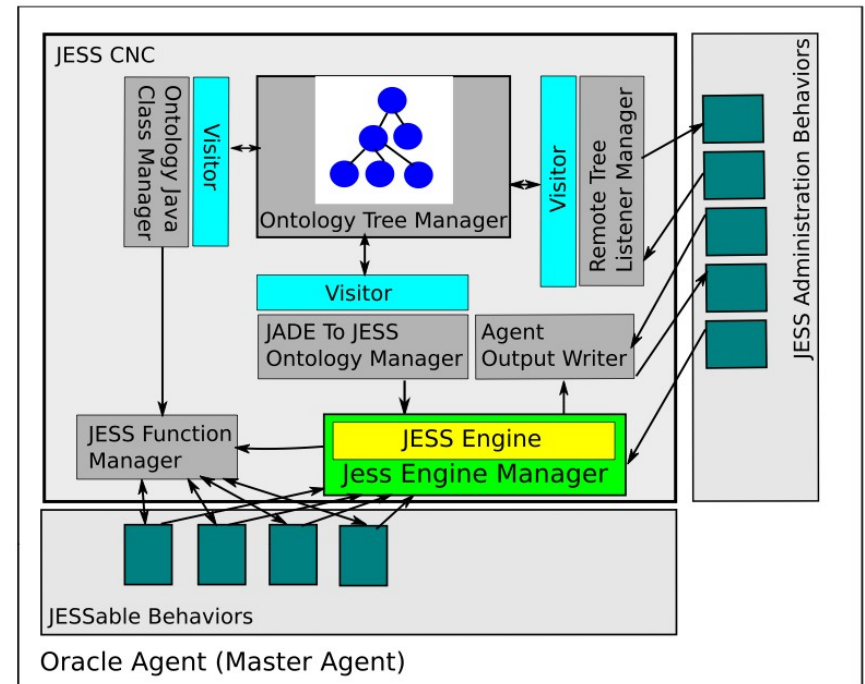
# Jess Remote Administration

- Remote “console” access to Jess.
- E.g.: add rules, manipulates facts
- Displays the domain ontology used by Jess
- Small Demo:

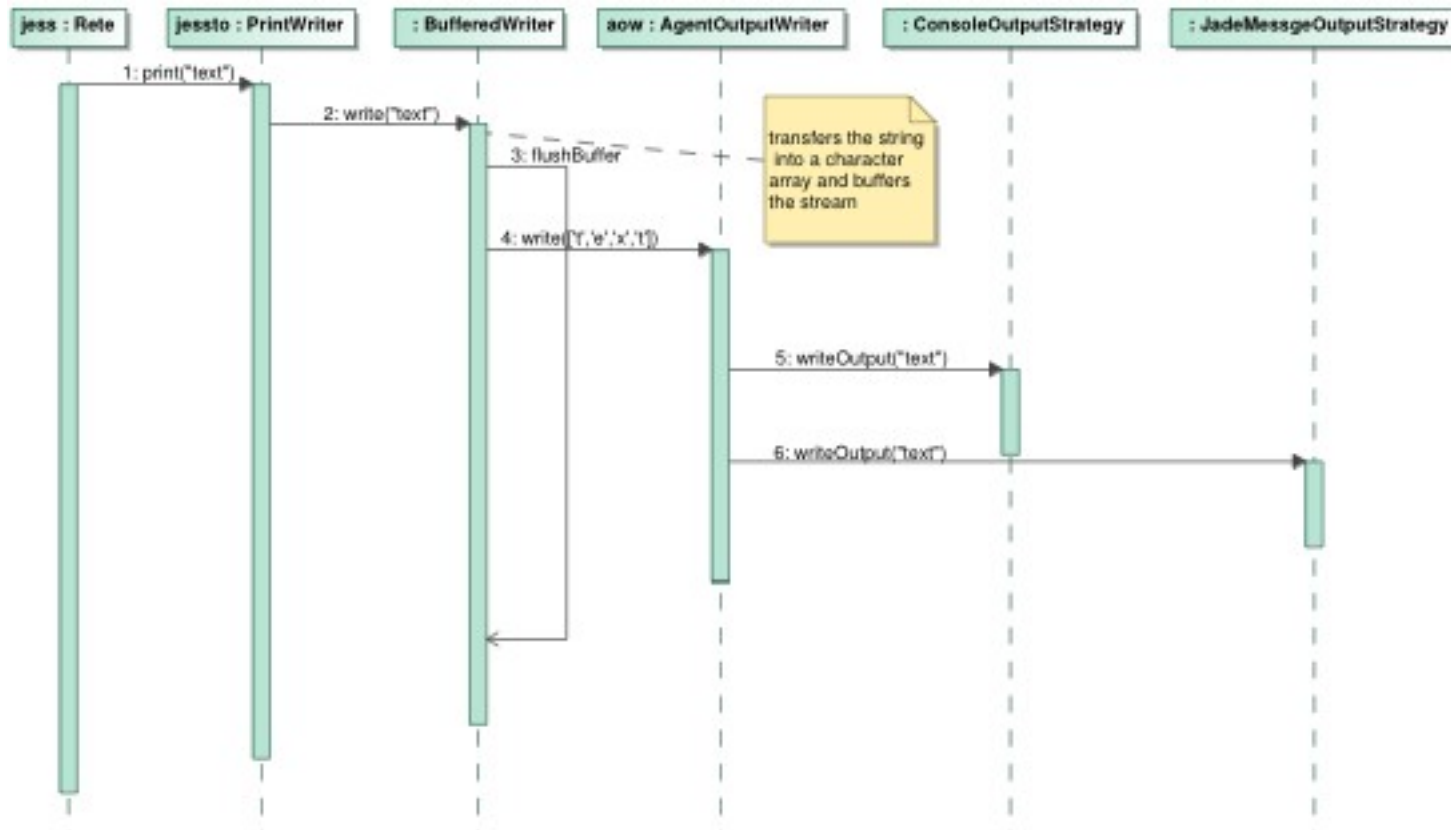


# The J2J Tool Kit

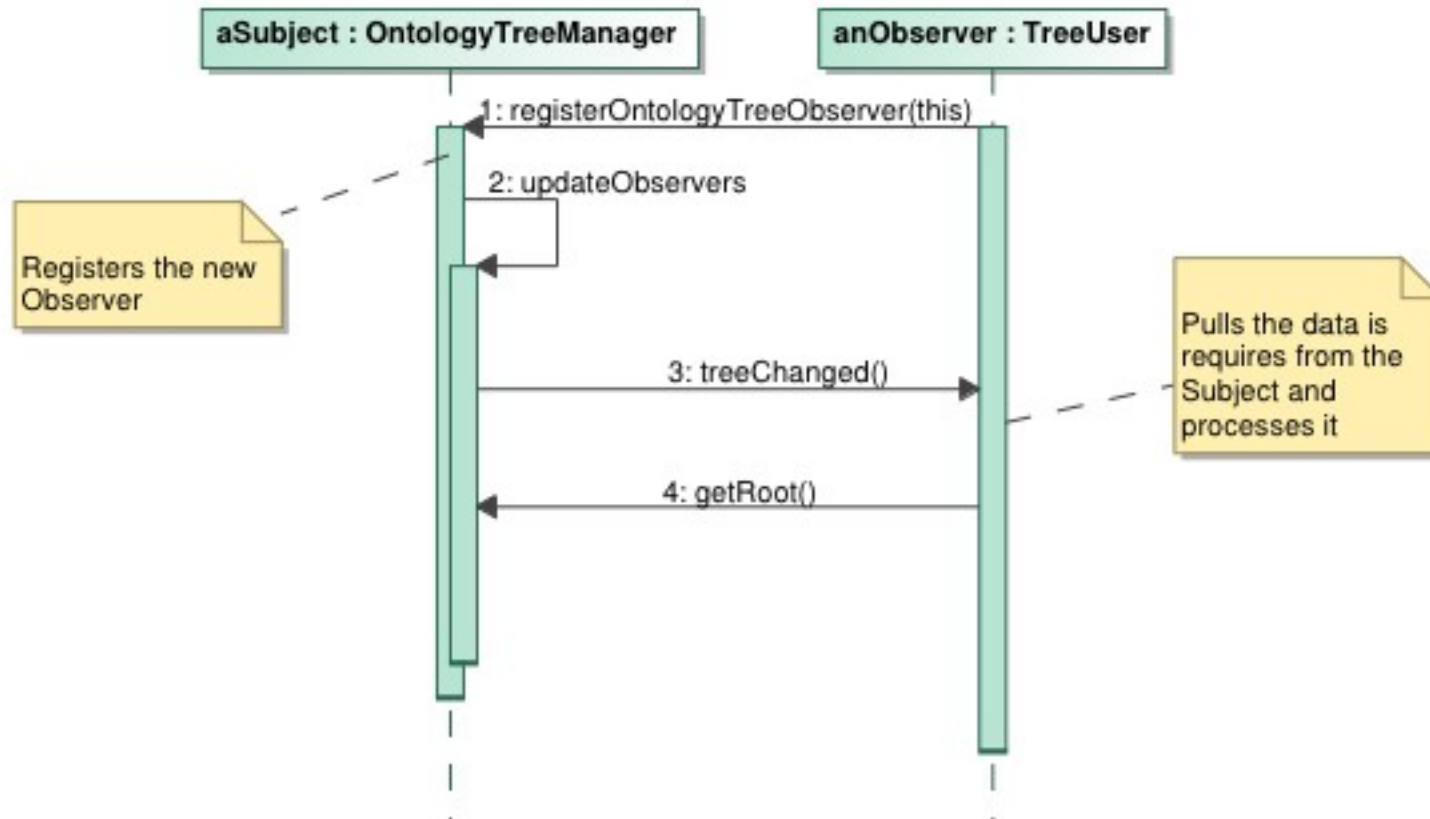
- JessCNC: the object that manages the integration.
- Two core elements:
  - The ontology
  - The rule engine



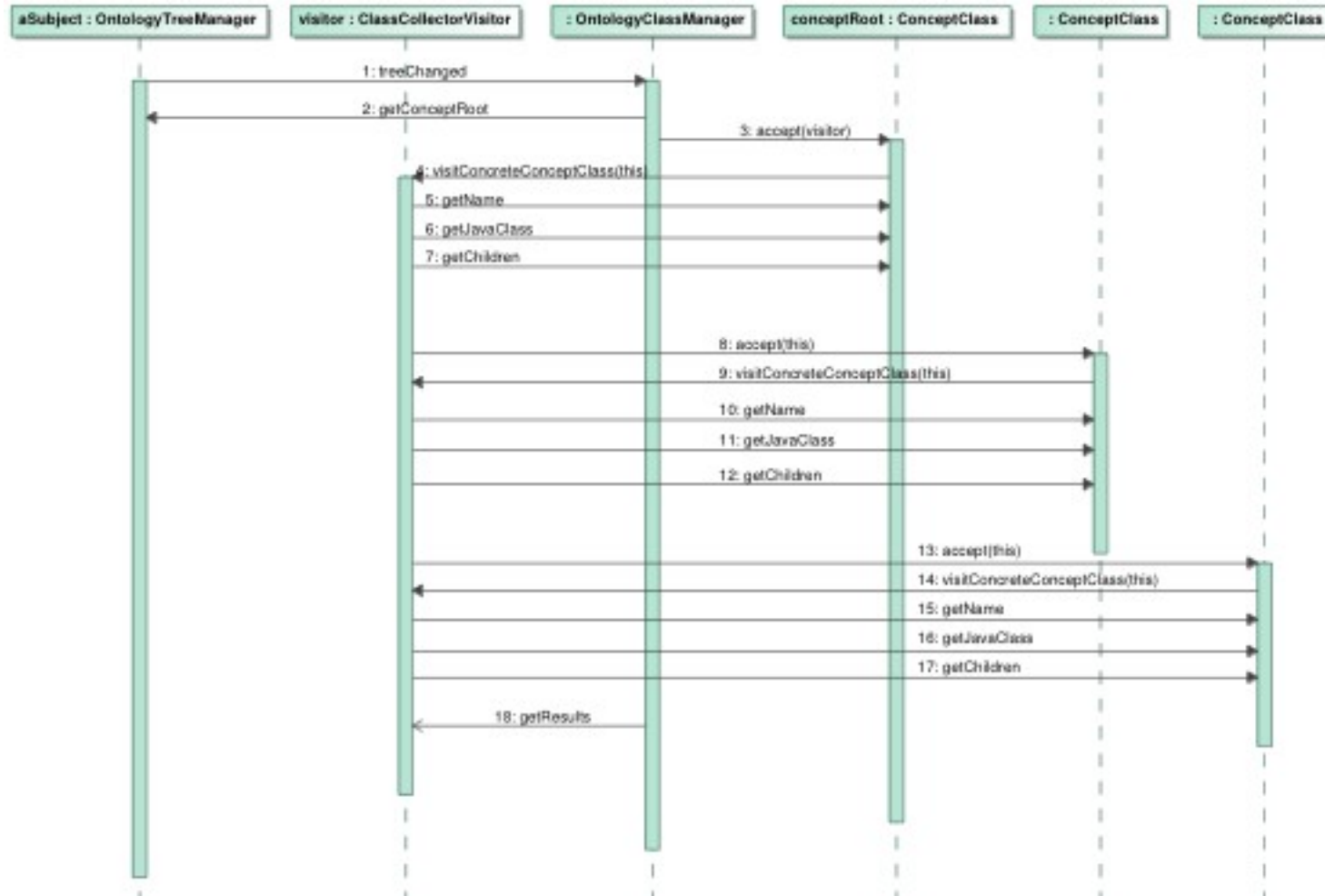
# Jess Output Routing



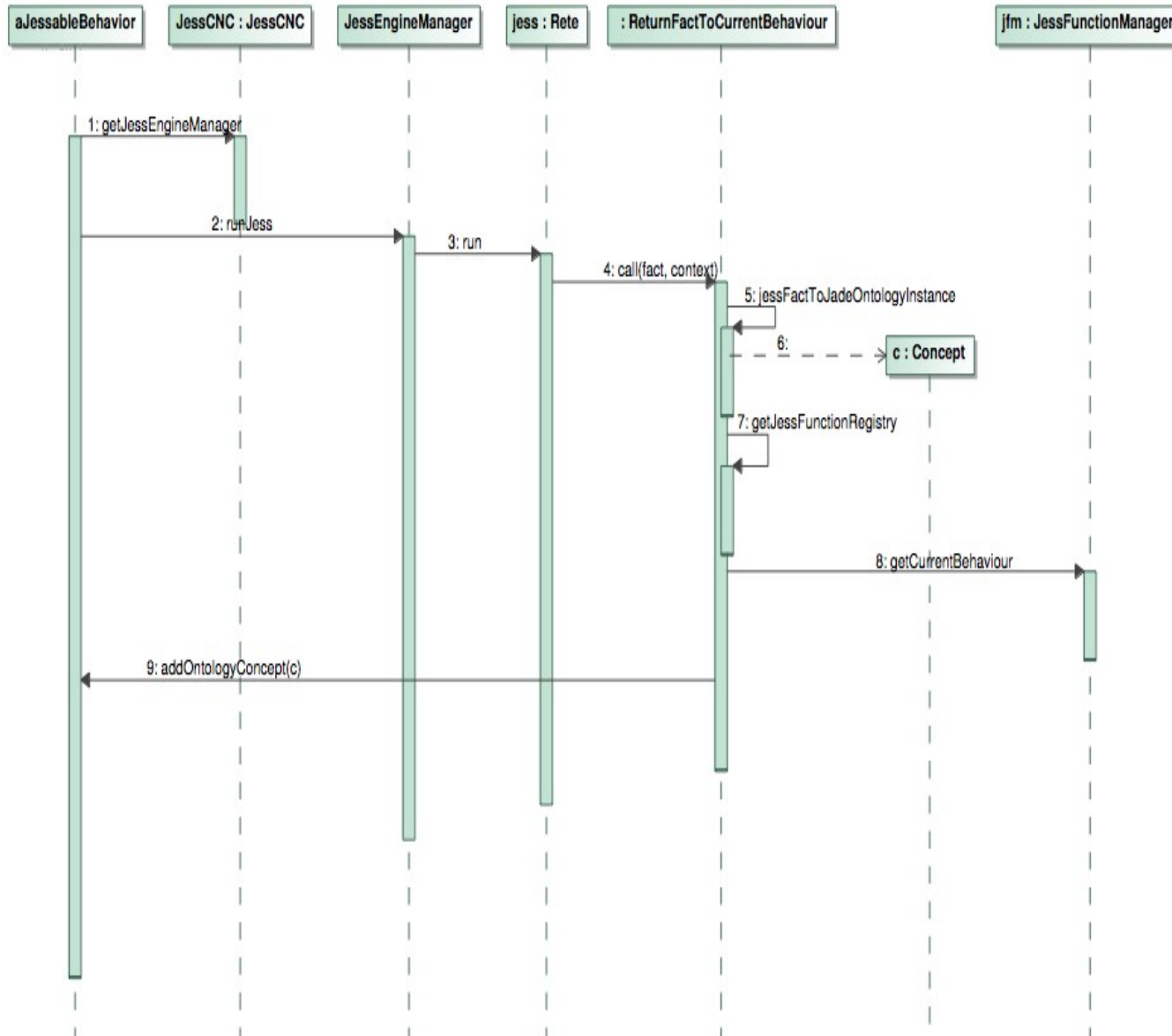
# Ontology Management::Update



# Ontology Management::Visitors



# Jessable Behaviors



# MediMAS Goes Jess

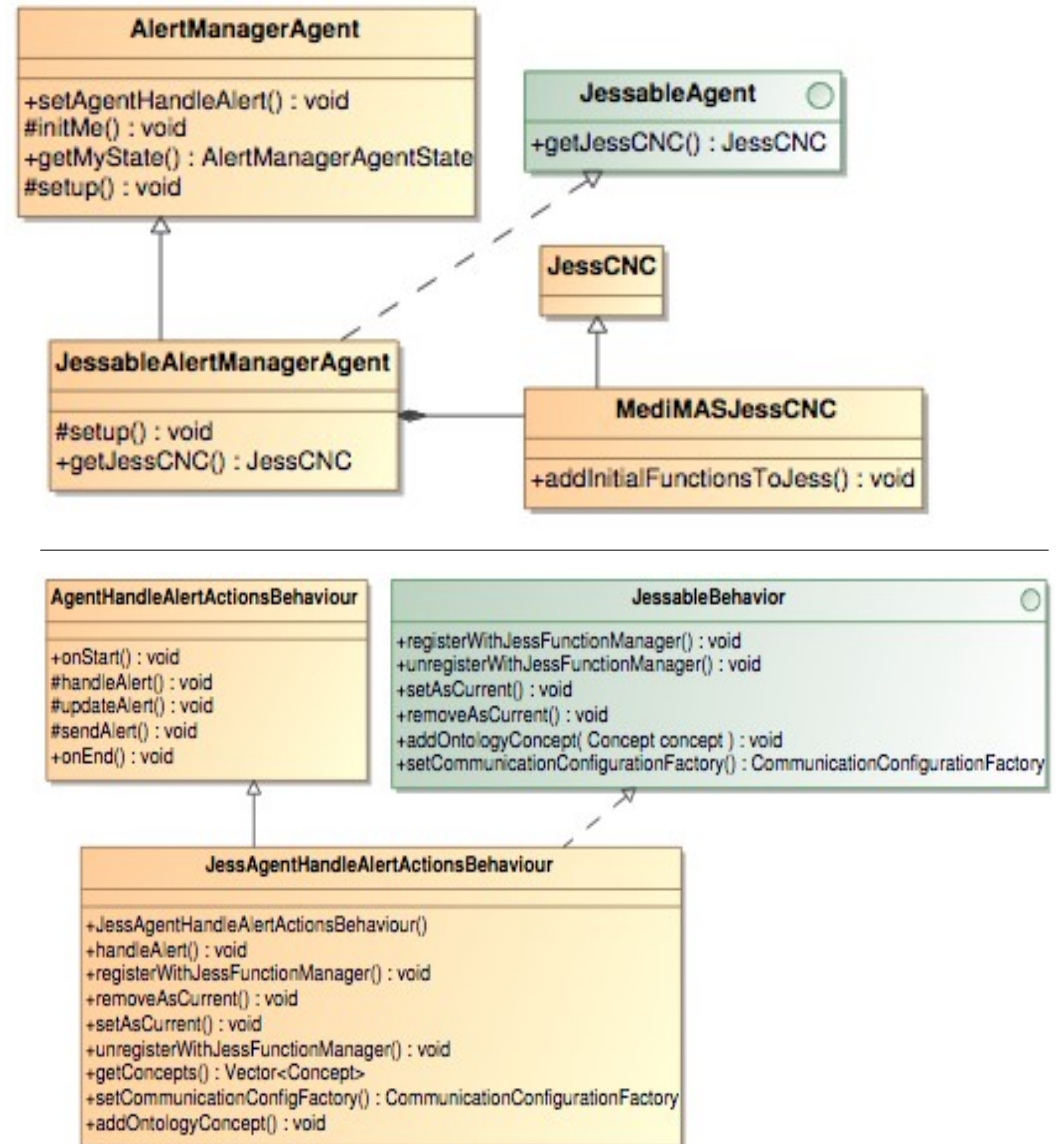
- Current state:
  - Stable solution, scales well.
  - But not flexible: Logic hard coded: it cannot be changed during run time.

```
protected void handleAlert( Concept concept ) {
2   AlertAction alert = (AlertAction)concept;
   long lastalerttime = alert.getLastAlertTime().getTime();
4   long currenttime = new Date().getTime();

   int intval = 1000 * HospitalConstants.NORMAL_PENDINGTIME ;
   if ( alert.getCritical() == true )
8     intval = 1000 * HospitalConstants.CRITICAL_PENDINGTIME ;
   else if ( alert.getUrgent() == true )
10    intval = 1000 * HospitalConstants.URGENT_PENDINGTIME ;
   if ( (currenttime - lastalerttime) >= intval ) {
12     // Update alert infos in the database
     this.updateAlert( alert );
14     // Send alert to requester / (and) labdirector
     this.sendAlert ( alert );
16   }
}
```

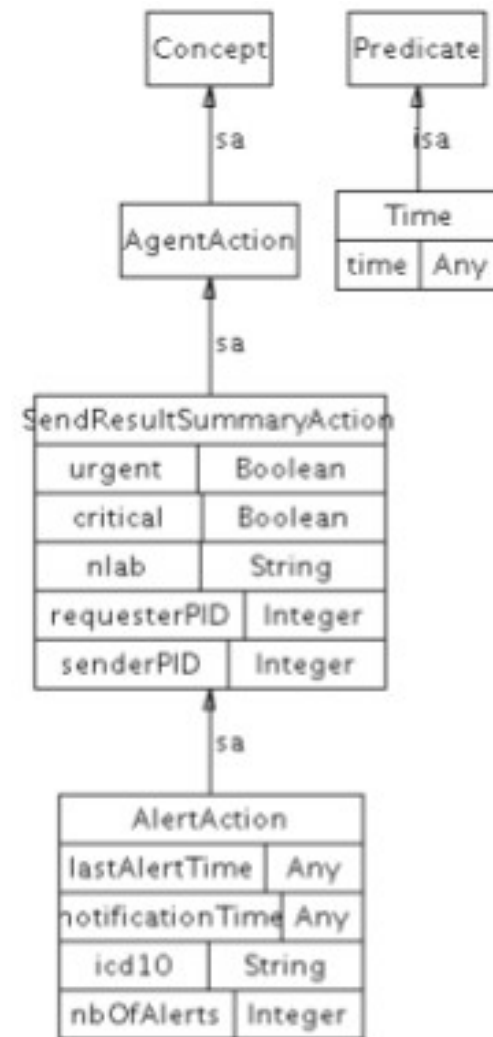
# MediMAS Goes Jess: Classes

- Approach:
  - Keep MediMAS, only replace notification decision making with Jess.
  - MediMAS classes are not altered but extended.
  - New classes use the J2J tool kit.



# MediMAS Goes Jess: Ontology

- One new concept:
  - Time: Represents a point in time.
- ICD-10 Field for Alert Action:
  - Domain ontology to classify alerts based on their medical content.



# MediMAS Goes Jess: Rules 1/2

```
(defrule MAIN::critical-alert
  (SystemTime (time ?t))
  ?a <- (AlertAction
         (critical TRUE)
         (nlab ?nlab)
         (senderPID ?sp)
         (requesterPID ?rp)
         (lastAlertTime ?l&:(>= (- (?t getTime) (?l getTime)) 10000))
         (nbOfAlerts ?nboa&:(> 7 ?nboa)))
  =>
  (modify ?a
   (nbOfAlerts (+ ?nboa 1))
   (lastAlertTime ?t))
  (printout t "Critical Alert NLAB " ?nlab " Sender " ?sp " Requester " ?rp crlf)
  (return-fact-to-current-behaviour ?a)
  (retract ?a))
```

```
(defrule MAIN::urgent-alert
  (SystemTime (time ?t))
  ?a <- (AlertAction
         (critical FALSE)
         (nlab ?nlab)
         (senderPID ?sp)
         (requesterPID ?rp)
         (lastAlertTime ?l&:(>= (- (?t getTime) (?l getTime)) 30000))
         (nbOfAlerts ?nboa&:(> 7 ?nboa)))
  =>
  (modify ?a
   (nbOfAlerts (+ ?nboa 1))
   (lastAlertTime ?t))
  (printout t "Urgent Alert NLAB " ?nlab " Sender " ?sp " Requester " ?rp crlf)
  (return-fact-to-current-behaviour ?a)
  (retract ?a))
```

# MediMAS Goes Jess: Rules 2/2

- Increase intervals in which Notification are treated:

```
(change-alert-check-time 100)
```

- Detect H5N1:

```
(defrule IMPORTANT::h5n1-found
  ?a <- (AlertAction (icd10 ?icd&:(= ?icd "J09"))
  (nlab ?nlab))
  =>
  (printout t "bird flue case located in lab order "
  ?nlab crlf "contact swiss ministry of health" crlf)
  (return-fact-to-current-behaviour ?a)
  (retract ?a))
```

# Conclusion 1/3

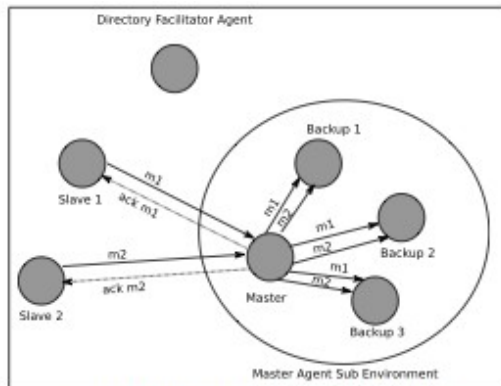
- Achievements:
  - Revision of eHealth processes
  - J2J tool kit:
    - The intelligent agent as service user
    - Interoperability
    - JADE services for Jess
    - Remote Jess Administration
    - Generic tool kit
    - Modular design

# Conclusion 2/3

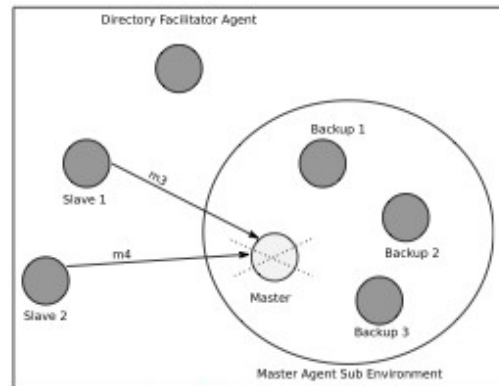
- Configuration File
- Automate Generation of “(agent action, behavior)” pairs
- Further development of the Remote Administration Agent
- Improved facts to object mapping
- Add Support for UMLS
- MediMAS with Multiple Intelligent Agents

# Conclusion 3/3

- Add support for backup agents



(a) Master agent operates normally



(b) Master agent fails

